Moderately Fast Data Collection

Up to One Kilohertz with an IRM

Mon, Mar 7, 1994

Introduction

The Internet Rack Monitor (IRM) includes hardware support for 1 KHz digitizing of all 64 analog channels, with possible expansion to 128 channels using two analog interface boards. This note describes a scheme for data request software support of this kind of data, so that it most easily fits into the Classic protocol. The scheme supports a data request for sampling digitized data at rates up to 1 KHz. The data points returned are time-stamped and can provide times relative to a selected Tevatron clock events or times relative to the time of the first data point returned in reply to the request.

Hardware

The hardware writes 64 channels of data each millisecond into a 64K byte circular buffer. This provides room for 512 sets of data, so the circular buffer wraps every half second. A new listype (#82 decimal) is supported for handling access to this data. The ident used with this listype is 3 words in length: the node#, the analog channel#, and an optional clock event specification. From the channel#, the software deduces the area of memory where the 64K buffer resides by adopting a channel assignment convention. We usually assign channels `0100–013F` for the first/only 64-channel block in an IRM. In case a second analog board is used, we assign channels `0140–017F` to the second 64-channel block. (Actually, any block of 64 channels that starts on a 64-channel boundary is ok, with bit#6 nonzero selecting the second block.) The first block uses memory addressed as `0063xxxx`, and the second block uses memory accessed at `0062xxxx`. The registers are based at `FFF58300` for the first block and `FFF58200` for the second block. The following table summarizes this:

Module	*Chan*	*Memory*	*I/O registers*
IP_d	`0100–013F`	`0063xxxx`	`FFF583xx`
IP_c	`0140–017F`	`0062xxxx`	`FFF582xx`

In order to activate the A/D hardware scanning, it is necessary to install a type `0x28` entry in the Data Access Table. The format of this entry is as follows:

```
2800      InitChan  I/O_register_base
—         —         Mem_base   #Chans
```

As an example,

```
2800      0100      FFF5       8300
0000      0000      0063       0040
```

Only the upper word of the memory base address is given, as its size is 64K bytes.
This example serves to copy all 64 readings from the most recent complete set of digitized data in the circular buffer memory into the data pool. The base address of this memory is `00630000`, and the I/O register base address is `FFF58300`. The presence of this entry in the Data Access Table enables the A/D scanning hardware logic. It also provides 15 Hz sampling in the data pool.

Request protocol

How is the sample period specified? It can be derived from the request reply period and the #bytes of data requested. The requester must provide enough buffer space to hold the data desired, which is central to the Classic protocol, anyway. Given the buffer size and the size of an 8-byte reply header and providing for 4 bytes per data point, the number of points that can be fit in the buffer is known. Given the time between successive replies, and assuming the rate of digitization is 1 KHz, the number of points available in the circular buffer memory is determined. The average time between points is then (#points_available)/(#points_in_buffer). Expressed as a delta set#, including a fractional part, it is used in the loop that maps the data points in the hardware circular buffer into the points to be placed into the user's buffer. Note that the scheme adapts to any variation in the time of updating the data request, because the point last copied is remembered between updates and therefore effectively measures the time since the last reply.

The reply format is then

time_of_first point (4 bytes)
time_between_last_two_events (4 bytes)
array of points as data, time (4*n bytes)

The format of the reply header is two longwords, occupying 8 bytes. The first longword is the time of the first data point in the reply buffer, expressed in units of 10 μs. The second longword is used only with the clock event# option. If this option is selected, by specifying a nonzero event# in the low byte of the third word of the ident used in the data request, then the second longword is the difference in time between the last two clock events of that kind, also expressed in units of 10 μs. It is needed to cover the case that the clock event occurred during the time represented by the reply data that follows. The requester should add each point's time value to the time_of_first_point to get the digitizing time of each point. If this time > delta time between the last two clock events, then reduce the time by this delta.

When a periodic request is received, the reply period is used to where in the circular buffer to begin sampling data points. If the period is one cycle, say, then the first reply provides data sampled over the one cycle period just *preceding* the request.

In case the clock event option is not used, the second longword is meaningless to the requester. But the first longword provides the time of the first data point relative to the time of the first data point in the first reply, which is one reply period *prior* to the time the data request is initialized.

The format of a data point is a pair of 16-bit integers. The first word is the data value in the usual two's complement left-adjusted fraction of full scale. The second word is the relative time since the starting time in the reply header, in units of 10 μs. The relative time value of the first data point in the reply = zero by definition.

Example

Suppose it is desired to collect 100 Hz samples from an analog channel. If the reply period is specified as 1 cycle, and the station runs at 15Hz, then 6–7 points should be collected every 15 Hz cycle. If the reply header is 8 bytes in length, then the requester should ask for 8+4*7=36 bytes of reply data. When it is time to update this request, the front end notices, for example, that 66 data sets have been stored in the hardware circular buffer since the previous update cycle about 1/15 second earlier. To perform the current update, 7 points will be selected out of 66 available. The delta set# is then 66/7=9.42857, on the average corresponding to about 106 Hz. By accumulating this value in the copying loop and taking the integer part each time to advance to the next set, the 66 points are sampled with 9–10 set spacing. On another update cycle, suppose that 70 data sets have been collected since the previous update cycle. Then the delta set# would be 70/7=10, with no fractional part. This would result in 10 set spacing, or 100 Hz exactly.

If the user specifies a larger buffer in the above case, say 8+4*70=288 bytes, then the calculated delta set# might be 66/70=0.94286, so some duplicate data values would appear in the reply data. In all cases, the reply buffer will be entirely filled. Since the circular buffer wraps every 512 ms, one should not specify a reply period more than 0.5 second (seven 15 Hz cycles or five 10 Hz cycles). Also, the largest buffer size necessary is 8+4*512=2056 bytes, since only the last 512 sets of 64-channel data are stored in the hardware circular memory buffer.

One-shot case

If a one-shot request is issued for such data supported by this listype, there is no reply period indicated. In this case, an attempt is made to reach back as far as possible to collect data to fill the requester's buffer. This might be a good time to supply a large buffer, in order to get the fullest time resolution available for later perusal. This facility allows for a host to monitor for some trip condition, say, and make a request that takes advantage of the hardware circular buffer memory to snapshot the last 0.5 seconds of a signal.